Supporting Ad-Hoc Queries in an Integrated Clinical Database

Sherry A. Steib[a], Richard M. Reichley[b], S. Troy McMullin[b], Keith A. Marrs[a], Thomas C. Bailey[a], Wm. Claiborne Dunagan[a], Michael G. Kahn[a]
[a]Washington University School of Medicine, St. Louis, Missouri
[b]The Jewish Hospital of St. Louis, St. Louis, Missouri

Whether caring for patients or conducting research, medical decision-makers need access to clinical data. To fulfill that need, commercial software developers have produced a wide range of database query tools that differ greatly in functionality and cost. Generally, tools that have a greater ability to conceal database complexity from the user also require more effort for administrative setup. We describe a cost-effective, commercially-available query tool that requires no special setup to perform most simple queries, yet can be customized to satisfy users' more complex querying requirements.

INTRODUCTION

BJC Health System is a large network of hospitals, outpatient/urgent care centers, private clinics, and long-term care facilities affiliated with the Washington University School of Medicine. In recent years, the school's Section on Medical Informatics has developed an integrated patient database which receives data from various hospital information systems.[1] These data include patient demographics, admissions/discharges/transfers, positive microbiology cultures, drug orders, drug allergies, dietary orders, and selected laboratory test results.

For various groups of medical decision-makers, Informatics researchers developed several non-interactive expert systems that utilize this database. After performing tasks such as microbiology culture surveillance[2] or verifying the appropriateness of patient drug orders[3], these systems store their recommendations in the database. Clinical information that is available for analysis, then, includes data that are integrated from multiple source systems and information generated by the expert systems. For the researcher, these data are invaluable in answering important, ad-hoc medical questions and validating the models upon which new applications depend.[4]

Researchers also developed personal computer (PC) applications with which the users could examine the expert systems' recommendations and enter additional related information. Equipped with a graphical user interface, each application was developed for a specific group of users focused on a small set of well-defined tasks. None of these PC applications is robust enough to support ad-hoc queries, so the users must request technical assistance in collecting any patient data necessary for analysis. Providing the users with tools for accessing and analyzing the database themselves would help to maximize the utilization of this valuable resource.

Some hospitals have developed elaborate tools for searching, displaying, and analyzing data.[5] Others have noted that the cost of integrating commercial software is small compared to the cost of custom software development and maintenance.[6] To minimize both the cost of initial development and on-going support, we chose to utilize a commercial query tool and to provide custom-developed supporting modules only where necessary. Structured Query Language (SQL) based tools such as *Microsoft Query*, Andyne's *GQL*, and ClearAccess Corporation's *ClearAccess* would all have been acceptable. Microsoft Query was chosen because it is both cost-effective and integrated with Microsoft Excel, the spreadsheet application our users prefer. This inter-operability enables the users to perform ad-hoc queries and then to easily import the query results into Excel for analysis.

METHODS

Hospital clinical pharmacists use PC applications such as word processors and spreadsheets in much of their work. To enable pharmacists to access and to analyze the patient data stored in a Sybase database on a remote server in the Informatics laboratory, we installed software on their PCs and provided a small set of custom-developed application programs for them.

A Novell TCP/IP product *(LAN WorkPlace)* supports the network connection between the two computers. Sybase's *Open Client Net-Library for LAN WorkPlace* enables PC applications to access the remote database. *Microsoft Excel for Windows* is used for data analysis. Included in the package with Excel is

Microsoft Query for Windows, an Open Database Connectivity (ODBC) compliant application that allows users to perform ad-hoc database queries. Query can be used either as a stand-alone program or in combination with Excel. When used with Excel, Query allows user-selected data to be returned directly into Excel for analysis. Query works with many different types of databases and requires a different ODBC driver program for each. Several Microsoft ODBC drivers are included with Excel. The *SQL Server Driver* was needed for our database.

With Microsoft Query, users have three different methods of performing queries:

1. Users can specify the query graphically, as shown in Figure 1. This method works well for simple queries and is the most frequently used method.
2. Users can enter and execute SQL programming language statements. Since the pharmacists are not SQL programmers, they do not write their own SQL programs. However, Informatics researchers can provide them with text files containing SQL statements, which the users can import into Query.
3. Users can execute *stored procedures*, which are SQL programs that are stored in the database. Stored procedures enable users to perform more complex queries than either of the other two methods. A small number of these procedures were developed by Informatics researchers.

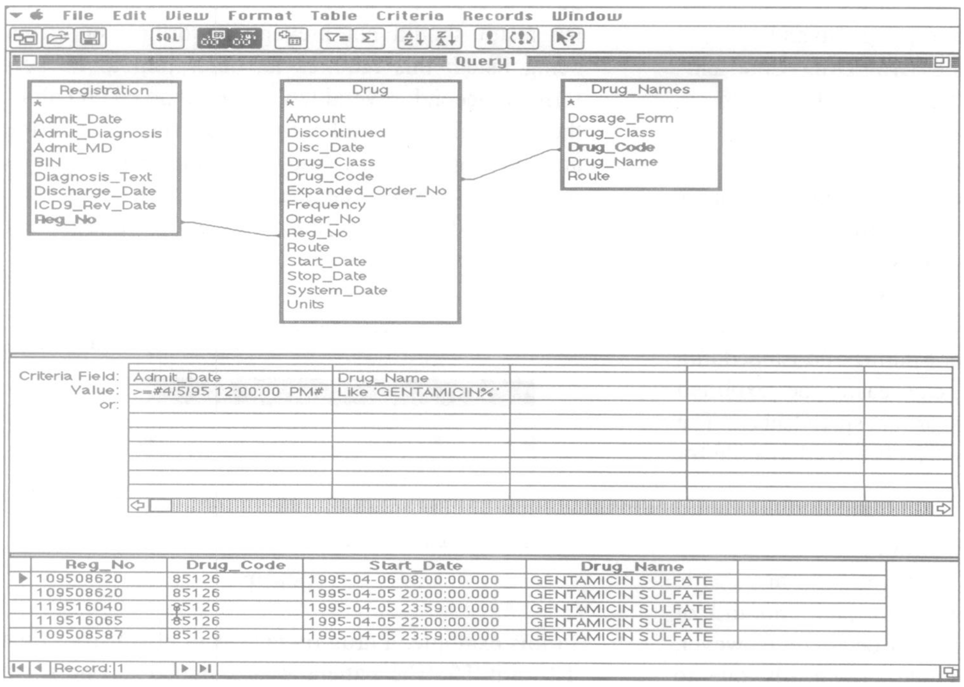
Regardless of which method of performing queries is used, it is important that the users have some understanding of the database structure. Although they were technically inclined and highly motivated, the pharmacists were not familiar with relational database concepts. To facilitate their learning process, we diagrammed the database schema, created a small number of views and stored procedures, and taught them how to use Query. To maintain data security, all access privileges to the tables, views, and stored procedures are controlled by the Sybase database administrator.

RESULTS

Many queries are relatively straightforward and can be performed using the first query method.

A graphical query example

Figure 1 shows a typical example of the output from this type of query. In the figure, the Query window is split horizontally into three "panes." The *Table pane* at the top shows which tables or views the user chose to include in the query. Each table is represented by a box with the table name at the top. The names of the columns in each table are listed below the table name; the user adds a column to the output by clicking its name. After choosing the tables and columns, the user must establish how the tables are to be joined. Lines connect the names of columns that are joined. Inner, outer, and self-joins are supported. Query automati-



Three window panes

Table pane
The user chose which tables (boxes) to be included in the query and how those tables were to be joined (lines).

Criteria pane
The user defined this criteria for selecting the rows.

Data pane
The query returned these rows of data.

Figure 1: A simple query performed with Microsoft Query

cally creates default join lines by inner joining columns that are primary keys in one table with columns in another table that have the same name and data type. The user can delete or change the default joins.

The middle pane, or *Criteria pane*, allows the user to define criteria for retrieving a subset of rows. Although the process of defining the criteria is not shown in the figure, it consists of creating one or more expressions. These expressions are combinations of operators, identifiers, functions, and literal values that evaluate to a single value. The rows that are returned by the query must satisfy all the conditions shown in the criteria pane. In the example, two expressions are shown: (1) Admit_Date must be after 12PM, 4/5/95, and (2) Drug_Name must start with 'GENTAMICIN'.

After choosing the tables and columns, drawing the join lines, defining the criteria, and specifying a sort order, the user can perform the query. Like many other data access tools, Query first converts the user's query specification into an SQL select statement before executing it. The SQL statement can be displayed, modified, or saved to a file--a useful debugging feature. In this example, the SQL statement is:

```
SELECT
    Registration.Reg_No,
    Drug.Drug_Code, Start_Date, Drug_Name
FROM
    Registration, Drug, Drug_Names
WHERE
    Drug.Reg_No=Registration.Reg_No and
    Drug_Names.Drug_Code=Drug.Drug_Code and
    Registration.Admit_Date>='4/5/95 12:00' and
    Drug_Name like 'GENTAMICIN%'
```

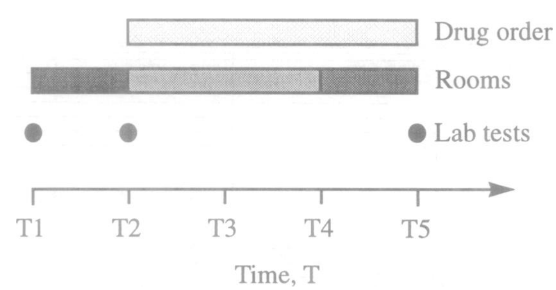
The *Data pane* at the bottom of the window displays the rows of data that the query returned: four selected columns of information about gentamicin orders for all patients admitted after 12PM, 4/5/95. Due to data-sharing schemes such as Microsoft's Dynamic Data Exchange (DDE) and Object Linking and Embedding (OLE), the query results can easily be exported directly to a word processor or spreadsheet. The graphical query specification can be saved to a file for later re-use.

A stored procedure query example

Query supports several different types of joins, a wide range of operators and expressions, and a number of other features that cannot be described here. However, all queries cannot be specified graphically due to Query limitations or restrictions. An outer join, for example, cannot be used in a query with more than two tables. The creation of subqueries, or "sub-select" statements, is not allowed. Complex nested conditional logic is also difficult, if not impossible, to specify graphically.

The second query method, whereby users enter (or import) queries as SQL statements, was intended for these much more difficult queries. This method allows the user to take full advantage of the power and expressiveness of the SQL language. Since the pharmacists are not SQL programmers and importing text files is cumbersome, this method is rarely used. Instead, we identified a number of these very difficult queries that users need to perform regularly and we implemented them as stored procedures. For maximum flexibility, the procedures were designed to accept parameters and to return most of the columns from any table used in the query. Although the stored procedures are all too long to be listed here, an example of the type of query they perform can be described.

Pharmacists typically need to know which patients are being given a particular drug. Sometimes, they are also interested in knowing the patient's room number and the result of any serum creatinine lab test that has been performed. When monitoring current patient drug orders, they are interested in up-to-date information. For a retrospective study, they are interested in older information. In either case, two aspects of this query cause it to be complex. First, the absence of a lab result should not prevent the drug order from being listed; this requires an outer join to be performed. Second, several tables are involved and all of them contain temporal information such as starting and ending dates. Complex conditional logic is required to ensure that the proper data is selected. Represented on a time-line, the data for one patient might appear as follows:



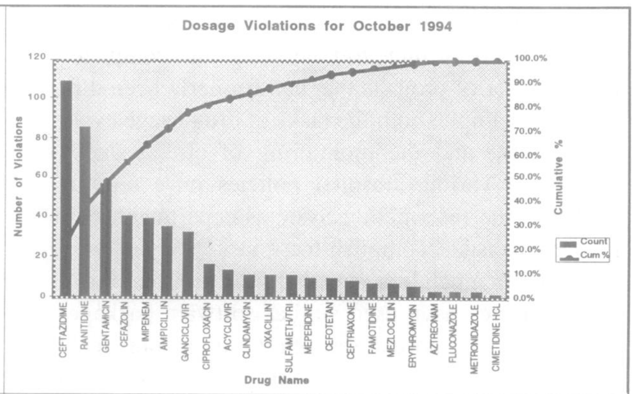
In this example, a drug order was in effect from time T2 until T5. The patient was assigned to one room from T1 until T2, to a second room from T2 until T4, and to a third room from T4 until T5. Three lab tests were performed at T1, T2, and T5. Performing a query that shows only the relevant information at time T3 is not a trivial task.

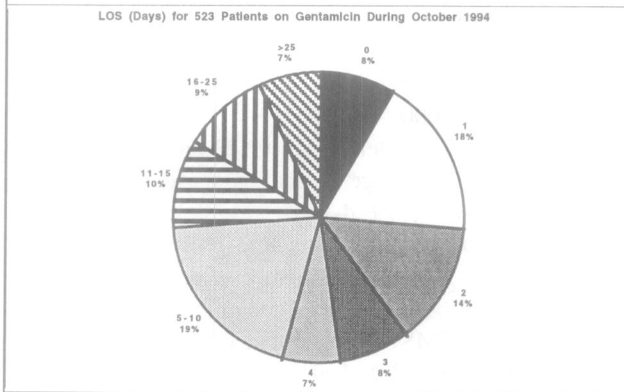
A much more difficult query returned the patient's maximum serum creatinine test results during various time intervals, as well as an indicator of whether the patient had an active order for several other drugs within the same time period. This type of query is difficult even for an experienced SQL programmer to specify and cannot be performed with a single select statement. Rather, a temporary table is usually created and a series of updates on it is performed. This allows one to split the complex conditional logic into smaller, more manageable pieces, overcome Query's outer join limitation, and perform subqueries where needed. To further simplify both the stored procedure and the users' graphical queries, a view was created which allowed serum creatinine test results to be selected from the lab test results table without specifying individual test and battery numbers.

DISCUSSION

With Excel, Query, and the database views and stored procedures that were created to support them, the valuable information contained in the integrated patient database is now accessible to the pharmacists. The charts in Figure 2 illustrate several types of anal-

A pharmacist analyzing output from the expert system *DoseChecker* can easily produce the Pareto diagram on the right, which shows dosage violations during October, 1994. Half the dosing errors were for ceftazidime, ranitidine, or gentamicin. This information can be used to focus attention on specific drugs with the greatest number of dosing errors.





The pie chart on the left shows length of stay (LOS) for patients given gentamicin during October, 1994. A little less than half the 523 patients were in the hospital longer than 4 days. Additional investigation would be performed to correlate the orders to the patient's diagnosis.

The stacked bar graph on the right shows the number of patient drug orders monitored by *DoseChecker* from 9/19/94 to 3/31/95. Plotted by medical service, the total number of orders for each service is easily compared to those orders (in black) containing potential dosing errors. This information can be used to educate appropriate medical staff.

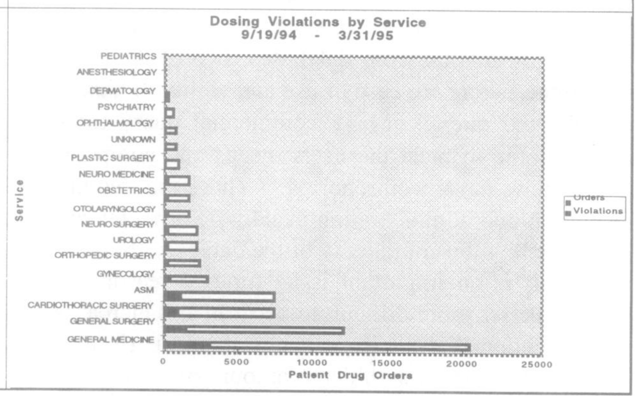


Figure 2: A few data analysis tasks that can now be performed by the pharmacy users

yses they are able to perform on data returned from their queries. At this writing, they are particularly interested in the output from an expert system named *DoseChecker*[3], which monitors patient drug orders and issues an alert when it detects an inappropriate dosage. Data for the top two charts are easily obtained from a query that the user specifies graphically; data for the bottom chart must be obtained by executing a stored procedure. After realizing the utility of the database, other users have begun to request similar access to it. To accommodate them, Excel and Query have been installed on additional PCs and Apple Macintoshes and a commercial natural language query interface is also being integrated.

With direct access to the data and tools for analyzing it, the pharmacists can facilitate decision-making, simplify many of their routine tasks, and complete a large number of projects that had formerly been difficult. For example, both the tasks of drug usage evaluation (DUE) and the monitoring of effects due to changes in various hospital policies once required help from the Information Systems department. Now the pharmacists accomplish these tasks without assistance, and in much less time than before. Other hospital pharmacies have used similar software packages, but none has reported the ability to automatically import ad-hoc query output into a spreadsheet for analysis. Rather, the data are either manually-entered or reside in ASCII text files created by mainframe programs that run at a specific time of day[7]. These files are then transferred from the mainframe to the PC and manually imported into a spreadsheet.[8,9]

Many factors must be considered when choosing a query tool: cost, vendor support, skills required of the user, effort to setup and administer, extent to which end-user access can be controlled, portability across computing platforms, kinds of databases that can be accessed, and degree of integration with other desktop software. Since a query tool's purpose is to help users extract information from databases, two of the most important factors are ease of use and ability to handle complicated queries. Many commercial query tools are available to meet the users' needs and comparative reviews have been reported.[10] Those that require less setup and administration usually do a poorer job of concealing the complexity of the database from the user. This is an important issue for two reasons. It makes queries more difficult to perform and increases the likelihood that a user will mistakenly perform a query that returns a significant amount of data--leading to a degradation of system performance for all users.

The temporal nature of clinical data mandates the use of complex conditional logic and outer joins for some queries. More frequently, however, simpler query logic is sufficient. One tool that meets all users' requirements is unlikely to be available in the near future. For now, experience shows that combining a well-rounded user interface that makes most queries easy to perform with customized software that simplifies the remainder is a good compromise.

ACKNOWLEDGMENTS

This work is supported by NLM Grant R29 LM05387, NLM Grant U01 LM05845, and funding from the Barnes Hospital Department of Pharmacy.

Reference

1. Marrs KA, Steib SA, Abrams CA, Kahn MG. Unifying heterogeneous distributed clinical data in a relational database. SCAMC 1993;17:644-648.
2. Kahn MG, Steib SA, Fraser VJ, Dunagan WC. An expert system for culture-based infection control surveillance. SCAMC 1993;17:171-175.
3. Abrams CA, Steib SA, Marrs KA, Jepson K, Kahn MG. An expert system for appropriately dosing renally excreted drugs. Submitted for publication 1995.
4. Stevens LE, Huff SM, Haug PJ. The development of a virtual database to provide on-line access to a large archive of clinical data. SCAMC 1992;16:600-604.
5. Safran C, Porter D, Rury CD, et. al. ClinQuery: Searching a large clinical database. MD Computing 1990;7(3):144-153.
6. Clayton PD, Anderson RK, Hill C, McCormack M. An initial assessment of the cost and utilization of the Integrated Academic Information System (IAIMS) at Columbia Presbyterian Medical Center. SCAMC 1991;15:109-113.
7. Mason RN, Pugh CB, Boyer SB, Stiening KK. Computerized documentation of pharmacists' interventions. Am J Hosp Pharm 1994;51:2131-2138.
8. Burnakis TG. Translating mainframe computer data to spreadsheet format. Am J Hosp Pharm 1991;48:2619-2622.
9. Smith SL, Lubiejewski T, Farnum J. Spreadsheet interface for transfer of drug-use data from a mainframe to a personal computer. Am J Hosp Pharm 1990;47:2488-2492.
10. Tyo, J. Query tools help users dip into data. InformationWeek 1995;April 10:54-62.